Contents

THEORY OF COMPUTATION (V-SEM., IT-BRANCH)

UNIT - I :

UNIT - II :

UNIT - III :

UNIT - IV :

UNIT - V :

UNIT 1

EORY OF COMPUTATION, DESCRIPTION OF FINITE TRANSITION FUNCTIONS, NING FINITE AUTOMATA, AUTOMATA, EQUIVALENCE ND MOORE MACHINES

f computation ?

nch of computer science that deals
ed on a model of computation using

(R.G.P.V., June 2010)

odel of a digital computer. Every
s. It has a mechanism for reading
g over a given alphabet, written on
omaton can only read. This file is
one symbol. The input mechanism
t, one symbol at a time. Also the
input string (sensing end-of-file

cells, each of which can hold a single symbol from an alphabet (not necessarily the same as the input alphabet). The automaton can read as well as change the contents of the storage cells. Finally, it has a *control unit*, which can be in any one of a

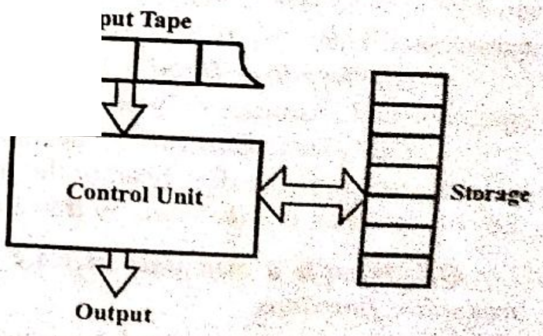


Fig. 1.1 A General Automaton

finite number of ***internal states.*** The control unit can change state in some defined manner. Fig. 1.1 shows a diagram of a general automaton.

An automaton is supposed to operate in a discrete time frame. At some given time, the control unit is in some internal state, and the input mechanism is scanning

a specific symbol on the input tape. The

step is obtained by the ***next-state*** or ***tra***

gives the next state in terms of the curr

input tape, and the information curren

transition from one time interval to th

information in the temporary storage ch

particular state of the control unit, input t

from one configuration to the next is ca

Q.3. Write short note on finite a

Ans. Finite automata is denoted l

(i) Q : finite set of states.

(ii) Σ: finite set of inputs.

(iii) δ: transition function w

(iv) q_0 : initial state, $q_0 \in Q$.

(v) F : set of final states suc

The above model is represented g

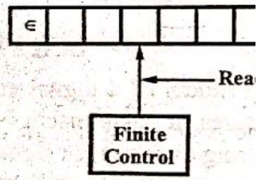


Fig. 1.2 Block Diagran

Various components of a finite au

(i) Input Tape – Input tape

squares, each square containing a sing

is processed from left to right in input

(ii) Reading Head – The h

input tape and examines only one squa

(iii) Finite Control – The fi

currently under the reading head or the

the output which is calculated by transition function on the inputs.

Q.4. What is a trap state in FA ? State and explain the properties of transition functions. ***(R.G.P.V., Dec. 2015)***

Ans. The automata in fig. 1.3 remains in its initial state q_0 until the first b is encountered. If this is also the last symbol of the input, then the string is accepted.

If not, the DFA goes into state q_2, from which it can never escape. Such a state is called a trap state.

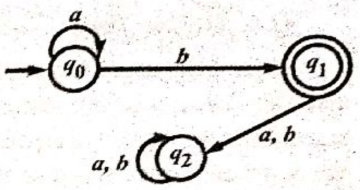


Fig. 1.3

The properties of transition functions are as follows –

naton.

anged

mbols a,

$\delta (q, a), w)$

$\delta (q, w), a)$

utomaton consumes or reads the

fter the automaton consumes a

h ? What are they ?

n graph or transition diagram, is

ta. It is just a simple graph with

enting transitions. The vertices

epresented by lines with arrows

ices are the names of the states,

out symbol values. For example,

n its associated graph will have

q_1) labelled a represents the

ented by an incoming unlabelled

nore, final states are shown by

F) is a deterministic finite

is exactly $|Q|$ vertices, each of

ch and every transition rule δ

led a. The vertex with label q_0

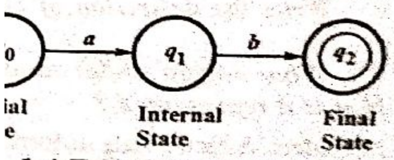


Fig. 1.4 Transition Graph

ponding transition graph and vice-versa. Fig. 1.4 represents the DFA,

$$M = (\{q_0, q_1, q_2\}, \{a,b\}, \delta, \{q_2\})$$

where δ is given by

$$\delta (q_0,a) = q_1$$
$$\delta (q_1,b) = q_2$$

Q.6. *Define finite state machine (FSM) with example.*

Ans. This machine will consist of input-output relation at every state and also the changes of the states that will occur on receiving the input at a particular state. Hence we will require the mapping in two forms. At a particular state, for a given input, what is the output.

$$S \times I \rightarrow O$$

and at a particular state on receiving the i

$$S \times I \rightarrow S$$

These are respectively known as ma
function (S.T.F.). Both of them are functio
and current input but their results are differ

(i) A finite set of states

$S = \{S_0, S_1, S_2,$

(ii) A special element of set 'S' s
(iii) An input alphabet, $I = \{i_0, i_1$
(iv) An output alphabet, $O = \{O$
(v) A function $S \times I \rightarrow S$ (S.T.F
(vi) A function $S \times I \rightarrow O$ (M.A.

At any state the machine is in any of i
symbol it will change the state using sta
output using machine function. As the nu
as finite state machine. The class of mach

Q.7. *Give a detailed explanation of*

Or

Discuss the finite state system with t

Or

Write a short note on one way finite

Or

Write the definition of DFA.

Or

What do you understand by DFA (D
how is it represented ?

Ans. A finite state automaton or finite automaton deterministic finite state machine (DFSM) or deterministic finite accepter (DFA) is a 5 tuple machine that can be represented as,

$$M = (Q, \Sigma, \delta, q_0, F)$$

where, (i) Q is a finite non-empty set of states

(ii) Σ is a finite non-empty set of input symbols called terminals or input alphabets

(iii) δ is a function which maps $Q \times \Sigma$ into Q and is usually called direct transition function. This function describes the change of states during the transition. Usually, this mapping is represented by a transition table or a transition diagram

(iv) q_0 is the initial state and $q_0 \in Q$

and $F \subseteq Q$.

rministic finite accepter is as follows –
itial state q_0, with its input mechanism
During every move of the automaton,
to the right, so each move consumes
g (i.e., after having all input symbols
is in one of its final states, otherwise
nly move from left to right, it cannot
ead only one input symbol at a time
n δ governs the transition from one
$(q_0, a) = q_1$, then, if initially the DFA
is a, the DFA will go into state q_1.

rministic finite automata.

(R.G.P.V., Dec. 2002)

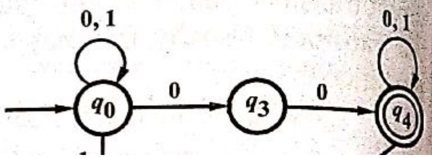
e of moves for an automaton. Instead
uation, this allows a set of possible
lefining the transition function such

DFA is defined as a 5-tuple machine –
$_0, F)$
f states
input symbols
napping from $Q \times \Sigma$ into 2^Q which
bsets of Q
Q
d $F \subseteq Q$.
d as,

$$\delta : Q \times (\Sigma \cup \{\Lambda\}) \rightarrow 2^Q$$

There are three major differences between DFA and NDFA. In an NDFA, the range of δ is in the power set 2^Q, so that its value is not a single symbol of Q, but a subset of it, which defines the set of all possible states that can be reached by the transition. For example, if the current state is q_1, the symbol read is a, and

then either q_0 or q_2 could be the next state of the NDFA. Also, we allow Λ as the second argument of δ. It means that the NDFA can make a transition without consuming an input symbol. Although in NDFA also, the input mechanism can only travel to the right it may become station
on some moves. Moreover, in an NDI
the transition $\delta(q_i, a)$ may be empty, i
there is no transition defined for t
particular situation.



Like DFAs, NDFAs can be represen
by transition graphs. The vertices can
determined by Q, while an edge (q_i,
with label b is in the graph if and only i
(q_i,b) contains q_j. But, since b may be
empty string, so there can be some ed
with labelled Λ or $\in$ or λ.

For example, consider the transitio

The sequence of states for the inp

An input string is accepted by an NDFA, if there is some sequence of possible moves that put the machine in a final state at the end of the string (i.e., after having the last symbol been read), and is rejected only when there is no possible sequence of moves by which any final state can be reached.

In the above example shown in fi
state, the input string 0100 will be accep
$\delta(q_0, 0100) = \{q_0, q_3, q_4\}$.

Q.9. Define two-way finite auton

O

Define two-way deterministic fin... automaton (2DFA).

Or

Write short note on two-way finite automata.

(R.G.P.V., Dec. 2002, June 2010)

Or

Define two-way finite automata.

(R.G.P.V., Dec. 2016)

Ans. The finite automaton is a control unit that reads a tape, moving one square right at each move. We added non-determinism to the model, which allow many "copies" of the control unit to exist and scan the tape simultaneously. Next we added Λ-transitions, which allowed change of state without reading
1. Another extension is to allow the
as right. Such a finite automaton is
FA. It accepts an input string if it
e tape, at the same time entering an
generalization does not increase the
accepts only regular sets. We give
-way FA, that is deterministic and
i.e., not stationary, at each move.
ıton or 2DFA is a quintuple
q_0, F)

the input alphabets

ınd
ıps from $Q \times \Sigma$ to $Q \times \{L, R\}$.
ınning input symbol a, the 2DFA
square. If $\delta(q, a) = (p, R)$, the
ht one square.

at behaves as follows, starting in
ıere in the tape head moves right
until encountering a '0', at which
epeated 'M' has three states, all of

1
q_1, R
q_2, L
q_2, L

and input string is $\boxed{101001}$.

Since q_0 is the initial state, the first ID is

$$\boxed{q_0 101001}$$

To obtain the second 'ID', note that the symbol to the immediate 'Right' of the state 'q_0' in the first 'ID' is a '1' and $\delta(q_0, 1)$ is (q_1, R) thus the second

ID is $\boxed{q_1,01001}$ continue in this way we get the result.

$q_0\ 101001$
$\xrightarrow[q_1,1,R]{}$
$\vdash 1q_1\ 0\ 1001$
$\xrightarrow[q_1,R]{}$
$\vdash 10\,q_1\ 1001$ move till we
$\xrightarrow[q_2,L]{}$
$\vdash 1q_2\ 01001$
$\xrightarrow[q_0,R]{}$
$\vdash 10q_0\ 1001$
$\xrightarrow[q_1,R]{}$
$\vdash 101q_1\ 001$
$\xrightarrow[q_1,R]{}$
$\vdash 1010q_1\ 01$
$\xrightarrow[q_1,R]{}$
$\vdash 10100q_1\ 1$
$\xleftarrow[q_2,L]{}$
$\vdash 1010q_2\ 01$
$\xrightarrow[q_0,R]{}$
$\vdash 10100q_0\ 1$
$\xrightarrow[q_1,R]{}$
$\vdash 101001q_1$

Q.10. Write short note on equivalen of DFA and NDFA.
(R.G.P.V., Dec. 2002, June 2010)

Ans. Since every DFA is an NDFA, it is clear that the class of languages accepted by NDFAs includes the regular sets (the language accepted by DFAs). However, it turns out that these are the only sets accepted by NDFAs. For every NDFA, we can construct an equivalent DFA that accepts the same language. The way a DFA simulates an NDFA is to allow the states of DFA to correspond to sets of states of the NDFA. The constructed DFA keeps track in its finite control of all states that the NDFA could be in after reading the same input as the DFA has read.

Two finite accepters M_1 and M_2 are said to be equivalent if

$$L(M_1) = L(M_2)$$

both accept the same languages.

shown in fig. 1.7. It is equivalent to
they both accept the same language.
nds with $ab\}$

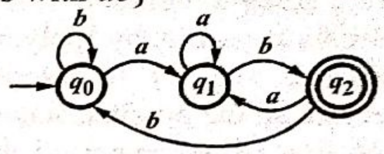


Fig. 1.8

rts an NDFA to DFA.

o DFA by following steps –
vertex $\{q_0\}$. Identify this vertex as
to e) until no more edges are missing –
$q_j,\ldots, q_k\}$ of graph G_D that has no
$(q_j, a),\ldots\ldots,\delta^*(q_k, a)$.
all these δ^*, yielding the set
h G_D labelled $\{q_l, q_m, \ldots\ldots, q_n\}$ if
h G_D from $\{q_i, q_j, \ldots\ldots, q_k\}$ to
whose label contains any $q_f \in F_N$
$\{q_0\}$ in graph G_D is also made a

Each pass through the loop in step (ii) adds an edge to graph G_D. But G_D has at most $2^{|Q_N|}|\Sigma|$edges, so that the loop eventually stops.

Q.12. Prove that for every NDFA, there exists a DFA which simulates the behaviour of NDFA.

Or

Prove that if L is the set accepted by NDFA, then there exists a DFA which also accepts L.

***Ans.* Proof** – Let $M_N = (Q, \Sigma, \delta, q_0, F)$ be an NDFA accepting L. We construct a DFA, M_D, as follows –

$$M_D = (Q', \Sigma, \delta', q_0', F')$$

where, (i) $Q' = 2^Q$ (any state in Q' is denoted by $[q_1, q_2, q_3, \ldots\ldots q_j]$

where $q_1, q_2, q_3, \ldots\ldots$

(ii) $q_0' = [q_0]$

(iii) F' is the set of all subse

Before going to define δ', we lool

M_N is initially at q_0. But, by applying an

of the states in $\delta(q_0, a)$. To describe M

a, we require all the possible states tha

a. So, M_D has to remember all these

Hence, the states of M_D are defined as

state q_0. q_0' is defined as $[q_0]$. A string

one of the possible states M_N reaches o

(i.e., an element of F') is any subset o

Now, we can define δ' as,

(iv) $\delta'([q_1, q_2, q_3, \ldots\ldots$

$= \delta(q_1, a) \cup \delta(q_2, a)$

Equivalently,

$\delta'([q_1, q_2, q_3, \ldots\ldots$

if and only if

$\delta(\{q_1, \ldots\ldots, q_i\}, a) =$

Before proving $L = L(M_D)$ we pro

$\delta'(q_0', x) = [q_1, \ldots\ldots$

if and only if $\delta(q_0, x) = \{q_1, \ldots\ldots,$

We prove by induction on $|x|$, the

$\delta'(q'_0, x) = [q_1, q_2, \ldots\ldots, q$

When $|x| = 0$, $\delta(q_0, \Lambda) =$

$\delta'(q_0', \Lambda) = q_0' = [q_0]$. So, equation (ii)

is a basis for induction. Assume that the

is only true for all strings

y with $|y| \le m$. Let x be a string of length $m + 1$. We may write x as ya, where $|y| = m$ and $a \in \Sigma$. Let $\delta(q_0, y) = \{p_1, \ldots\ldots, p_j\}$ and $\delta(q_0, ya) = \{r_1, r_2, r_3, \ldots\ldots r_k\}$. As $|y| \le m$, then by inductive assumption, we have –

$$\delta'(q_0', y) = [p_1, \ldots\ldots\ldots p_j] \qquad \ldots\ldots\text{(iii)}$$

Also, $\{r_1, r_2, r_3, \ldots\ldots\ldots, r_k\}$

$= \delta(q_0, ya)$

$= \delta(\delta(q_0\ y), a) = \delta(\{p_1, \ldots\ldots\ldots p_j\}, a)$

By definition of δ' –

$$\delta'([p_1, \ldots\ldots\ldots p_j], a) = [r_1, r_2, \ldots\ldots\ldots, r_k] \qquad \ldots\text{(iv)}$$

Hence,

$\delta'(q_0', ya) = \delta'(\delta'(q'_0, y), a)$

$\ldots\ldots, p_j], a) = [r_1, r_2, \ldots, r_k]$

[by equation (iii) and equation (iv)]

$q_3, \ldots\ldots\ldots q_i]$,

gs.

proved similarly.

x) contains a state of F. As we have

state of F if and only if $\delta'(q_0', x)$ is

$\in L(M_D)$. The proves that DFA,

outputs. What are the differences

hine ?

example) –

machine.

(R.G.P.V., June 2010)

utput. **(R.G.P.V., Dec. 2004)**

re machine. **(R.G.P.V., Nov. 2018)**

output, i.e., they accept the string

bility is decided on the basis of

tial state. This restriction can be

the output can be chosen from

ct approaches, first in which the

lled a ***Moore machine*** and second

ansition, called a ***Mealy machine.***

In other words, the value of the output function $Z(t)$ in the general case, is a function of the current state $q(t)$ and the present input $a(t)$, i.e.,

$$Z(t) = \lambda(q(t), a(t))$$

where λ is called the output function. This generalized model is called ***Mealy machine***. If the output function $Z(t)$ depends only on the present state and is independent of the current input, the output function may be written as –

This restricted model is called ***Moore machine***. It is more convenient to use Moore machine in automata theory.

Formally, we define each as follows –

(i) Moore Machine – The Moore machine is a six-tuple machine $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$, where

(a) Q is a finite set of

(b) Σ is the input alpha

(c) Δ is the output alph

(d) δ is the transition f

(e) λ is the output func

(f) q_0 is the initial state

The λ is a mapping from Q to Δ gi

The output of M in response

$\lambda(q_0)\lambda(q_1)$.......$\lambda(q_n)$, where q_0, q_1,

$(q_{i-1}, a_i) = q_i$ for $1 \le i \le n$. It is no

response to input Λ. The DFA may be v

where the output alphabet is {0,1} and state q is "accepting" if and only if $\lambda(q) = 1$.

For example, table 1.1 shows a Moore machine. The initial state q_0 is marked with an arrow. The table defines δ and λ. For the input string 0111, the transition of states is given by

$$q_0 \rightarrow q_3 \rightarrow q_0 \rightarrow q_1 \rightarrow q_2$$

The output string is 00010. For th

(ii) Mealy Machine – The M

$(Q, \Sigma, \Delta, \delta, \lambda, q_0)$, where all the symb

in the Moore machine. λ is the outpu

$\lambda(q, a)$ gives the output associated wi

The output of M in respons

$\lambda(q_0, a_1)\lambda(q_1, a_2)$,.........$\lambda(q_{n-1}, a_n)$, where q_0, q_1, ...,q_n is the sequence of states such that $\delta(q_{i-1}, a_i) = q_i$ for $1 \le i \le n$. It is noted that this sequence has length n rather than length $n + 1$ as for the Moore machine, and on input Λ a Mealy machine gives output Λ.

For example, table 1.2 shows a Mealy machine which defines δ and λ. For the input string 0011, the transition of states is given by

$$q_1 \rightarrow q_3 \rightarrow q_2 \rightarrow q_4 \rightarrow q_3,$$

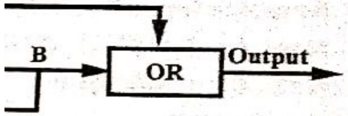
and the output string is 0100. In the case of Mealy machine, we get an output only on the application of an input symbol. So for input string Λ, the output is only Λ, while in the Moore machine the output is $\lambda(q_0)$,

Table 1.2 A Mealy Machine

Next State		
	$a = 1$	
put	*State*	*Output*
	q_2	0
	q_4	0
	q_1	1
	q_3	0

with output capability ? Draw a

g circuit.



(R.G.P.V., Dec. 2015)

o current by 0. We identify four

or not.

put	
	B
	0
	1
1	0
q_3 1	1

The operation of this circuit is such that after an input of 0's and 1's, the states changes according to the following rules –

New B = Old A

New A = (Input) NAND (old A OR old B) (∵ D Flip flop)

Output = Input OR old B

Suppose we are in state q_0 and we are not receiving current i.e., input = 0

New B = Old A = 0 (Intable, A = 0, in state q_0)

$\boxed{\text{New } B = 0}$

New A = (Input) NAND (old A OR old B)

New A = 0 NAND (0 OR 0)

New A = 0 NAND 0

$\boxed{\text{New } A = 1}$

Output = Input OR old B

Output = 0 or 0

$\boxed{\text{Output} = 0}$

The next state is q_2 because, new

If we are in state q_0 and we recei

$\boxed{\text{New } B = \text{Old } A = 0}$

New A = Input NAND (old A

New A = 1 NAND (0 OR 0)

New A = 1 NAND 0

$\boxed{\text{New } A = 1}$

Output = 1 OR 0

$\boxed{\text{Output} = 1}$

The next state is q_2 (because new

If we are in state q_1

Let, Input = 0 (No current)

$\boxed{\text{New } B = \text{Old } A = 0}$

New A = Input NAND (old A OR

New A = 0 NAND 1

$\boxed{\text{New } A = 1}$

Output = 0 OR 1

$\boxed{\text{Output} = 1}$

therefore next state is q_2 ($\because A = 1, B = 0$)

when Input = 1 (with current)

$\boxed{\text{New } B = \text{Old } A = 0}$

New A = 1 NAND (0 OR 1)

New A = 1 NAND 1

$\boxed{\text{New } A = 0}$

Output = 1 OR 1

$\boxed{\text{Output} = 1}$

(from table)

$\boxed{\text{New } A = 1}$

Output = 0 OR 1

$\boxed{\text{Output} = 1}$

therefore next state is q_3 ($\because A = 1, B = 1$)

when input = 1

$\boxed{\text{New } B = \text{Old } A = 1}$

New $A = 1$ NAND (1 OR 1)

New $A = 1$ NAND 1

$\boxed{\text{New } A = 0}$

Output = 1 OR 1

$\boxed{\text{Output} = 1}$

therefore next state is q_1 $(\because A = 0, B =$

Table 1.3 Tra

Old State	Input =	
	New State	O
q_0	q_2	
q_1	q_2	
q_2	q_3	
q_3	q_3	

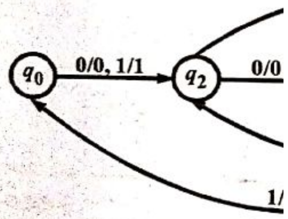


Fig. 1.10 Trans

Q.15. Explain the procedure for t Moore machine with the help of an e

Ans. Consider the Mealy machine in table. 1.4.

Table 1.4 An Exampl

Present State	Input $a = 0$		Input $a = 1$	
	State	Output	State	Output
$\rightarrow q_1$	q_3	0	q_2	0
q_2	q_1	1	q_4	0
q_3	q_2	1	q_1	1
q_4	q_4	1	q_3	0

The construction of the Moore machine which is equivalent to the Mealy machine is as follows –

At the first stage, we develop the procedure so that both machines accept exactly the same set of input sequences. We look into the next state column for any state, say q_i, and determine the number of different outputs associated

states, the number of such states

uts associated with q_i.

iated with one output 1 and q_2 is

. Similarly q_3 and q_4 are associated

, we split q_2 and q_4 into q_{20} and

can be reconstructed for the new

State Table

t State	
Input $a = 1$	
State	Output
q_{20}	0
q_{40}	0
q_{40}	0
q_1	1
q_3	0
q_3	0

state column can be rearranged

q_{21}	q_1	q_{40}	1
q_3	q_{21}	q_1	0
q_{40}	q_{41}	q_3	0
q_{41}	q_{41}	q_3	1

able 1.7 An Equivalent Moore Machine

resent State	Next State		Output
	$a = 0$	$a = 1$	
q_0	q_3	q_{20}	0
q_1	q_3	q_{20}	1
q_{20}	q_1	q_{40}	0
q_{21}	q_1	q_{40}	1
q_3	q_{21}	q_1	0
q_{40}	q_{41}	q_3	0
q_{41}	q_{41}	q_3	1

Table 1.6 gives the Moore machine. Here, we observe that the initial state q_1 is associated with output 1. This means th

of 1, if the machine starts at state q_1. Thus this Moore machine accep zero-length sequence (null sequence) which is not accepted by the Me machine. To overcome this situation, either we must neglect the response Moore machine to input Λ, or we must add a new starting state q_0, wh state transitions are identical with those of q_1 but whose output is 0 table 1.6 is transformed as ta

It is clear from the pr machine, the corresponding n states.

Q.16. Explain the proc
Mealy machine with the he

Ans. The following proc machine from a given Moor

(i) We have to de a function of present state ar

$$\lambda'(q,a) = \lambda(\delta$$

(ii) The transition machine.

Table 1.8 An Example of Moore Machine

Present State	*Next State*		*Out*
	a = 0	*a = 1*	
→ q_0	q_3	q_1	
q_1	q_1	q_2	1
q_2	q_2	q_3	
q_3	q_3	q_0	

Consider the Moore mac table 1.8.

We must follow the reve Moore machine. In case of form the pair consisting of reconstruct the table for Mealy machine.

For example, in table 1.8 the states q_3 and q_1 in the next state colu should be associated with outputs 0 and 1, respectively. The transition ta for Mealy machine is given in table 1.9.

It is noted that the number of states can be reduced in any model considering states with identical transition. If two states have identical transiti

i.e., the rows corresponding to the two-states are identical), then we can remove one of them.

Q.17. What is Moore machine ? How finite automata can be converted into Moore machine ? Explain with the help of example.

(R.G.P.V., June 2015)

Moore machine by introducing utput function $\lambda(q)$, such that

$\in F$

$\notin F$

ı by table 1.10.

e 1.11 Moore Machine

Next State		*Output*
a = 0	*x = 1*	
q_2	q_1	1
q_3	q_0	0
q_0	q_3	0
q_1	q_2	0

The value of output is defined

$\lambda(q_3) = 0$,

iting output values calculated hine is shown in table 1.11.

LEMS

strings containing four 1's in

(R.G.P.V., Dec. 2008)

$_5\}, \{0, 1\}, \delta, q_0, \{q_4\})$

The finite automaton is as shown in fig. 1.11.

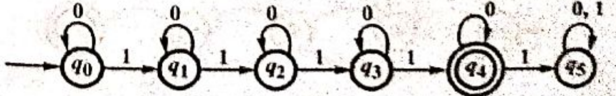


Fig. 1.11 Finite Automaton

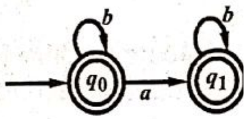
Imp

Prob.2. Design DFA that accepts all strings with at most 3 a's.
(R.G.P.V., June 20

Sol. The finite automaton M is given by

$$M = (\{q_0, q_1, q_2, q_3, q_4\}, (a, b), \delta, q_0, \{q_0, q_1, q_2,$$

The finite automaton is shown in fig. 1.12



Fi

Prob.3. Construct a finite a
$\{0^n \mid n \bmod 3 = 2,$

Sol. No. of strings generate
are –

00, 00000, 00000000,00000

So, finite automata for the giv
shown in fig. 1.13.

Prob.4. Design a NFA for {c

Sol. NFA for the language

$$L = \{cbab^n \mid n$$

is $\quad M = (\{q_0, q_1,$

Here, the NFA is such that i
cbabb, cbabbb, ...

So, NFA for the given langua

Prob.5. Draw a deterministi
the language {w/w contains the s

Sol. Given language is,

L = {w/w contains the substring abab}

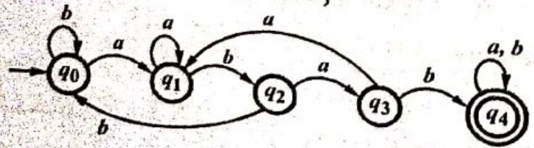


Fig. 1.15 Finite Automata Accepting {a, b}* {abab}

Prob.6. Design a machine which checks whether a given decimal number is "EVEN".

Sol. There will be two states one for EVEN and second one for odd.
0, 2, 4, 6, 8 indicate that they are even numbers because when we
r '0' otherwise odd number.

able

1, 3, 5, 7, 9
q_1
q_1
q_1

ven state is starting state i.e.,

e.

6, 7, 8, 9}

states

3, 5, 7, 9
0
0
0

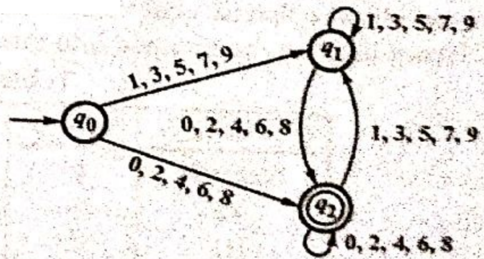
2 $\underline{q_2}$ 46710321

24 $\underline{q_2}$ 6710321

246 $\underline{q_2}$ 710321

2467 $\underline{q_1}$ 10321

24671 $\underline{q_1}$ 0321



$246710\ \underline{q_2}\ 321$

$2467103\ \underline{q_1}\ 21$

$24671032\ \underline{q_2}\ 1$

$246710\ 321\ \square$

As we end up in the s

number is odd.

Prob.7. Design a mach

by '4'.

Sol. To check divisibil

'0', remainder '1', remainc

starting state.

Q \ Σ
S
Rem '0' q_0
Rem '1' q_1
Rem '2' q_2
Rem '3' q_3

Note – We can merge t

and both of them are final

In table 1.14 row-q_1 a

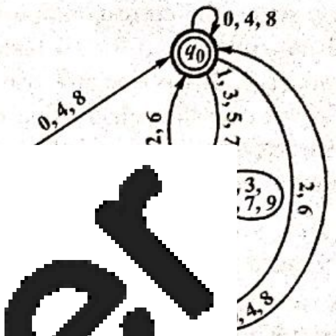
non-final states. So we are

Q \ Σ
Start
q_0
q_{13}
q_2

we can observe that two co

7. Hence they can be grou

Q \ Σ	0, 4, 8	1, 3, 5, 7, 9	2, 6
Start (S)	q_0	q_{13}	q_2
q_0	q_0	q_{13}	q_2
q_{13}	q_2	q_{13}	q_0
q_2	q_0	q_{13}	q_2



pter equivalent to M_N =

te table given in table 1.17.

M_N

i.e., ϕ, $[q_0]$, $[q_0, q_1]$, $[q_1]$.

as they are the only states

n table 1.18.

M_D

$[q_1]$	$[q_1]$	$[q_0, q_1]$
$[q_0, q_1]$	$[q_0, q_1]$	$[q_0, q_1]$

q_0 and q_1 appear in the rows corresponding to q_0 and q_1 and the column corresponding to a. So, $\delta([q_0, q_1], a) = [q_0, q_1]$.

When M_N has n states, the corresponding finite accepter has 2^n st However, we need not construct δ for all these states, but only for th states reachable from $[q_0]$, this is because, we have to construct only accepting $L(M_N)$. So, we start the construction of δ for $[q_0]$. We proceed considering only states appearing earlier under input columns and constru δ for them. We halt when nc

Prob.9. Give DFA ac
that all strings of 0 and 1

Sol. The expression (0
in 101. Now we construct a
corresponding to given lang

Fig. 1.18 shows a trans

Fig. 1.1

Now, we construct a tr

Table 1

Now, we convert above

Table 1.

Stat
$[q_1]$
$[q_1\ q_2$
$[q_1\ q_3$
$[q_1\ q_2\ q$

State diagram for requir
shown in fig. 1.19.

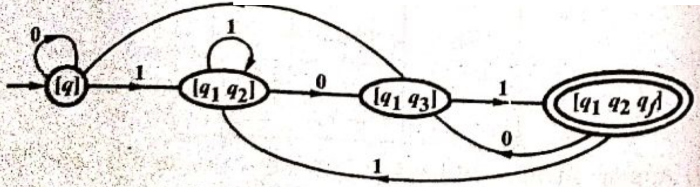


Fig. 1.19 DFA Accepting Strings Ending in 101

Prob.10. Design deterministic finite automaton accepting the following languages over the alphabet {0, 1} –

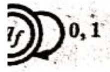
(i) The set of all words ending in 00.
(ii) The set of all words except ∈.
with 0.

(R.G.P.V., Dec. 2015)

iotes all strings of 0's and 1's
gram for the DFA.

ords except ∈ over the alphabet

words that begin with 0 over



uage over the alphabet 0, 1
(R.G.P.V., Dec. 2016)

Prob.12. Design FA which accepts even no. of 0's and even no. of 1's.
(R.G.P.V., Dec. 2011)

Or

Construct DFA over input alphabet Σ = {0, 1} to accept string which contains no. of 0 is even and no. of 1 is even. ***(R.G.P.V., June 2010)***

Or

Construct a DFA accepting set of all strings containing even no. of and even no. of b's over alphabet {a, b}. **(R.G.P.V., June 20**

Sol. Consider the transition diagram of fig. 1.23. This FA is denoted $M = (Q, \Sigma, \delta, q_0, F)$, where $Q = \{q_0, q_1, q_2, q_3\}$, $\Sigma = \{a, b\}$, $F = \{q_0\}$ δ is shown in fig. 1.24.

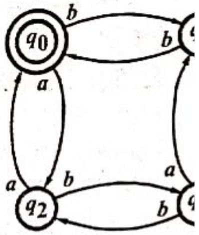


Fig. 1.23 The Transitio
of a Finite Auton

Suppose *bbabab* is input

$\delta(q_0, bb) = \delta(\delta(q_0, b), \text{b}) =$

Therefore, *bb* is in *L*(*M*),

$\delta(q_0, a) = q_2$. Thus,

$$\delta\,(q_0, b$$

Continuing in this mann

$$\delta\,(q_0, bb$$

and finally $\delta\,(q_0, bbab$

The entire state of seque

$$\overset{b}{q_0}\ \overset{b}{q_1}\ \overset{a}{q_0}\ \overset{b}{q_2}\ q$$

Thus *bbabab* is in *L*(*M*).

Prob.13. Construct a fin

(i) Having odd nu

(ii) Having even n

Sol. (i) Having odd nu

The finite automaton *M* i

$$M = (\{q_0, q_1\},$$

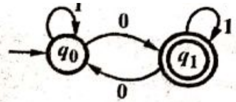


Fig. 1.25

(ii) Having even number of 0's and even number of 1's – Refe Prob.12.

Prob.14. Construct DFA for –

(i) Binary integer divisible by 3 (ii) $a^n b \mid n \geq 0$.

(R.G.P.V., Dec. 2009)

Sol. (i) In binary number when we to find the corresponding value

Table 1.21

Q Σ	*0*	*1*
→Ⓢ	q_0	q_1
'0') ◎q_0	q_0	q_1
'1') q_1	q_2	q_0
'2') q_2	q_1	q_2

bit

bit)

'0', i.e. state q_0)

bit)

bit)

it)

ate 'q_0'

next bit)

1' i.e., q_1)

1(next bit)

2' i.e., state q_2)

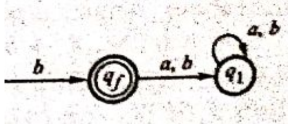


Fig. 1.26

Fig. 1.27

(ii) No. of strings generated by language $a^n b | n \geq 0$ are– *b*, *ab*, *aab*, ab, *aaaab*, ...

DFA for the given language is shown in fig. 1.27.

Prob.15. Construct a finite automaton that will accept those strings a binary number that are divisible by three. ***(R.G.P.V., Dec. 20***

Sol. Refer to Prob.14 (i).

Prob.16. Define deterministic finite automata. Draw DFA that acc any string which ends with 1 or it ends with an even number of 0's the last 1. Alphabets are {

Sol. **DFA –** Refer to Q

A *DFA* can be determi

$M = (\{q_0, q_1, q$

where δ is defined in form

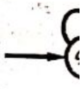
Class of language acc

$L = \{0, 1\}^* \{1$

Prob.17. Construct a the number of 1's in w is

Sol. The finite automa

$M = (\{q_0,$



Prob.18. Construct N

$S \to Ab|ab, A \to$

Sol. The above gramm of the form $\{a^n b^m \mid n, m \geq$ When the above grammar is

$S \to bA|ba, A \to$

Let $M = (Q, \Sigma, \delta[S],$ $[B], [aB], [\in]\}, T = \{a, b\}$

$\delta([S], \in) = [bA], [ba]$

$\delta([A], \in) = [bA], [bB]$

$\delta([B], \in) = [aB], [a]$

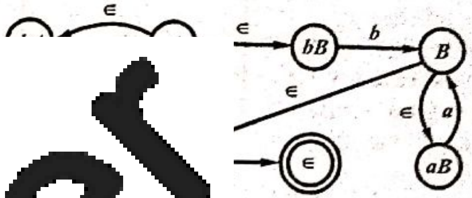
$\delta([bA], b) = [A]$

$\delta([ba], b) = [a]$

$\delta([a], a) = [\in]$

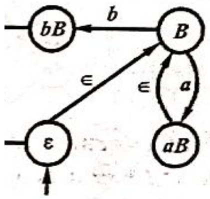
$\delta([bB], b) = [B]$

$\delta([aB], a) = [B]$



ear Grammar

wn in fig. 1.30. To get the final everse the NFA and the result



r Grammar

he set of all string over the ible by 5 and number of 1's ***(R.G.P.V., Dec. 2012)***

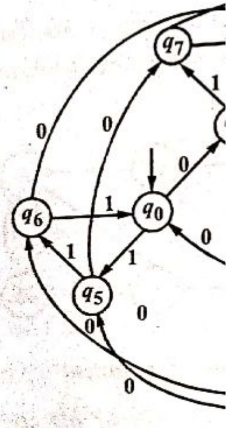
that number of zero divisible ***(R.G.P.V., Dec. 2013)***

1.32. This FA is denoted by $q_4, q_5, q_6, q_7, q_8, q_9, q_{10},$ d δ is shown in table 1.23.

able

		1
		q_5 First One
Second Zero q_1	q_2	q_7 First One
Third Zero q_2	q_3	q_9 First One
Fourth Zero q_3	q_4	q_{11} First One
Fifth Zero q_4	q_0	q_{13} First One
Second One q_5	q_7	q_6

	q_8	q_0
Third One q_6	q_9	q_8
Second One q_7	q_{10}	q_1
Third One q_8	q_{11}	q_{10}
Second One q_9	q_{12}	q_2
Third One q_{10}	[illegible]	q_{12}
Second One q_{11}		
Third One q_{12}		
Second One q_{13}		
Third One q_{14}		



Prob.20. Construct DFA

$M = (\{p, q, r, s\}, \{0,$

where δ is defined in the follo

Sol. The DFA M_1 equivalent to M is defined as follows –

$$M_1 = \{2^Q, \{0, 1\}, \delta, p, F\}$$

where, $F = \{[q, s], [p, q, s], [r, q, s], [p, q, r, s]\}$

We start the construction by considering $[p]$ first. The state table is in table 1.24.

Table 1.24 State Table of DFA

State	*0*	*1*
$[p]$	$[q,s]$	$[q]$
$[q]$	$[r]$	$[q,r]$
$[q,s]$	$[r]$	$[q,r,p]$
$[r]$	$[s]$	$[p]$
[illegible]	[illegible]	$[q,r,p]$
[illegible]	[illegible]	$[p,q,r]$
[illegible]	[illegible]	$[p]$
[illegible]	[illegible]	$[p]$
[illegible]	[illegible]	$[p,q,r]$

t to an NDFA whose transition

b
$_2,q_3$
q_3
q_2
–

(R.G.P.V., Dec. 2017)

valent to M is defined as follows –

able 1.25 State Table of DFA

State	*a*	*b*
$[q_0]$	$[q_1,q_3]$	$[q_2,q_3]$
$[q_1,q_3]$	$[q_1]$	$[q_3]$
$[q_1]$	$[q_1]$	$[q_3]$
$[q_2,q_3]$	$[q_3]$	$[q_2]$
$[q_2]$	$[q_3]$	$[q_2]$
$[q_3]$	ϕ	ϕ

to an equivalent deterministic

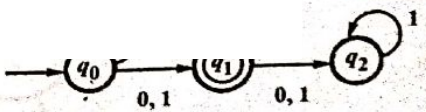


Fig. 1.33

Sol. The *NFA* starts in state q_0, so the initial state of the *DFA* wil labelled $\{q_0\}$. Since $\delta_N(q_0,0) = \{q_0, q_1\}$, we introduce the state $\{q_0, q_1\}$ in

G_D and add an edge labeled 0 between $\{q_0\}$ and $\{q_0, q_1\}$. In the same considering $\delta_N(q_0,1) = \{q_1\}$ gives us the new state $\{q_1\}$ and an edge label 1 between it and $\{q_0\}$.

Similarly,

$$\delta_N(\{q_1\}, 0) = \{q_2\}$$

$$\delta_N(\{q_1\}, 1) = \{q_2\}$$

$$\delta_N(\{q_0, q_1\}, 0) = \delta(\{q_0$$

$$= \{q_0,$$

$$\delta_N(\{q_0, q_1, q_2\}, 0) = \delta(\{q$$

$$= \{q_0,$$

$$\delta_N(\{q_0, q_1, q_2\}, 1) = \{q_1,q$$

$$\delta_N(\{q_0, q_1,\}, 1) = \delta(\{q$$

$$= \{q_1\}$$

$$\delta_N(\{q_1, q_2\}, 0) = \{q_2\}$$

$$\delta_N(\{q_1, q_2\}, 1) = \{q_2\}$$

$$\delta_N(\{q_2\}, 0) = \phi$$

$$\delta_N(\{q_2\}, 1) = \{q_2\}$$

The result, shown in fig. 1.34,

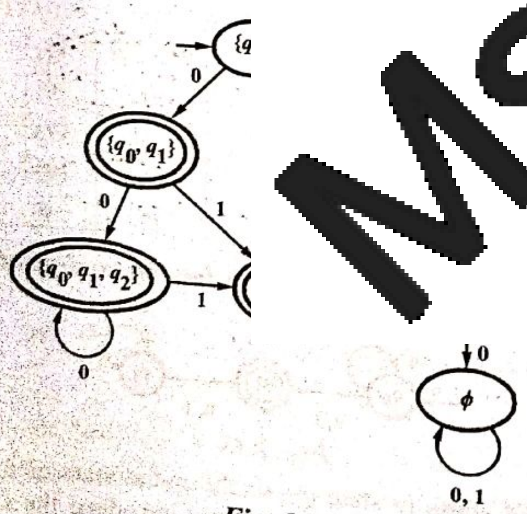
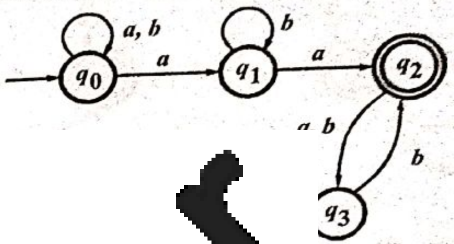


Fig. 1.34

Prob.23. ***Construct minimized DFA for the given NFA.***



(R.G.P.V., June 2009)

fig. 1.35). (R.G.P.V., Dec. 2008)

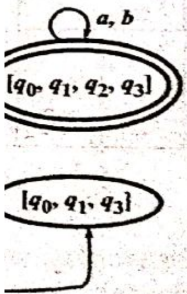
hows the transition table of the

Transition Table for the DFA

Input	
a	***b***
$[q_0, q_1]$	$[q_0]$
$[q_0, q_1, q_2]$	$[q_0, q_1]$
$q_0, q_1, q_2, q_3]$	$[q_0, q_1, q_3]$
$q_0, q_1, q_2, q_3]$	$[q_0, q_1, q_2, q_3]$
$[q_0, q_1, q_2]$	$[q_0, q_1, q_2]$

DFA, and given in table 1.27.

duced DFA.



Prob.24. ***Convert the following NFA with ∈ to equivalent DFA.***

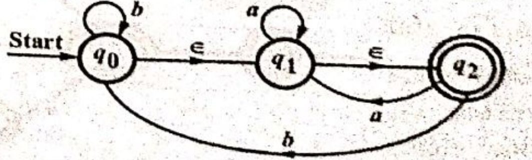
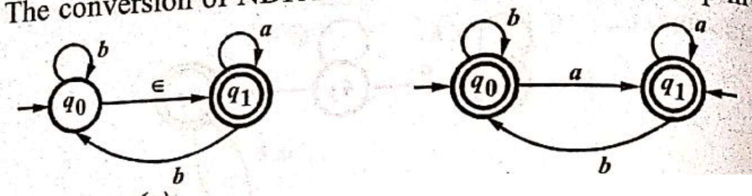


Fig. 1.37

(R.G.P.V., June 2011)

Sol. The conversion of NDFA to DFA is given step-by-step in fig. 1.



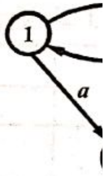
(a)

Fi

The DFA shown in fig. 1.38 (b

Fi

Prob.25. Convert the following



F

Sol. We construct the transition
∈-moves as given in table 1.28.

Table 1.28

State	Input	
	a	*b*
①	1, 3	–
②	1	–
3	2	3, 2

Ta

The successor table is constructe

The state diagram for the successor table is the required DFA as indicated in fig. 1.41 since ① and ② are the only final states of NDFA, therefore all states are the final states of DFA.

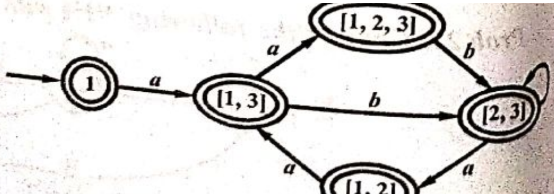


Fig. 1.41 Constructed DFA

Prob.26. Determine the DFA equivalent to the given NFA –

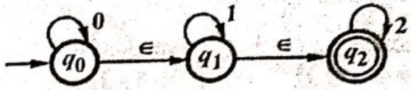


Fig. 1.42 ***(R.G.P.V., Dec. 2016)***

p such that there is a path from q

$, q_2\}$

$\}$

nd $\{q_2\} = A_3$

$))$

$) \cup \hat{\delta}(q_2, \in), 0))$

$(\because A_3 \subseteq A_2 \subseteq A_1)$

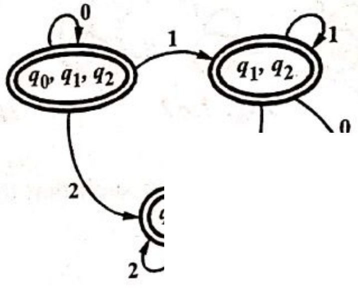
$\delta(q_2, 0))$

Transition matrix is shown in table 1.30.

Table 1.30

Input / State	0	1	2
q_0, q_1, q_2	q_0, q_1, q_2	q_1, q_2	q_2
q_1, q_2	ϕ	q_1, q_2	q_2
q_2	ϕ	ϕ	q_2

So the required DFA is shown in fig. 1.43.



Prob.27. Draw a Mealy ma
string is identical to the input str

Sol. The Mealy machine is gi

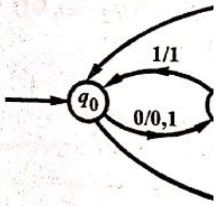


Fig. 1.44 Mea

Let us convert the transition
problem. Table 1.31 shows the re

Table 1.31 Mea

Present State	
q_0	q
q_1	q
q_2	q
q_3	q

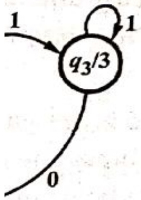
q_2 1

Prob.28. Construct a Moore machine which calculates mod-4 for eac binary string treated as binary integer.
(R.G.P.V., Dec. 200

Sol. Since last two digits of every binary integer decides whether integ is divisible by 4 or not. So that it can also decide what will be the remainder the integer is divisible by 4 as

Last Two Digits of Binary Integer	*Remainder*
00	0
01	1
10	2
11	3

d-4 for each binary string treated

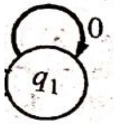


od-4 for Each Binary Integer

hich can output EVEN, ODD
tered is even or odd. The input
(R.G.P.V., Dec. 2010)

denotes even number of ones
construct F.S.M.

0	1
q_0	q_1
q_1	q_0



put-symbol.

MEALY MACHINE

We can check that given string containing even numbers of ones or odd numbers of ones. Suppose input string is → 01110

$\underline{q_0}$ 1 1 1 0 ⇒ O/P → EVEN

0 $\underline{q_0}$ 1 1 1 0 ⇒ O/P → ODD

0 1 q_1 1 1 0 ⇒ O/P → EVEN

0 1 1 q_0 1 0 ⇒ O/P → ODD

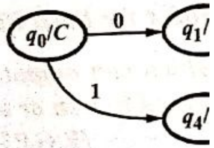
0 1 1 1 q_0 0 ⇒ O/P → ODD

0 1 1 1 0 $\boxed{q_1}$

q_1 shows that given string conta

Prob.30. Give Mealy and M
the input ends in '000', outpu
otherwise output C.

Sol. This machine is design
the output is *A*, if the input ends
partial design of it can be made



Now, we will insert the po
the Moore machine becomes

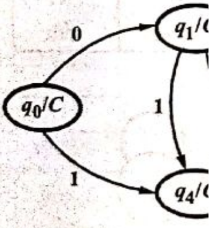


Fig. 1.

Now the Mealy machine wi

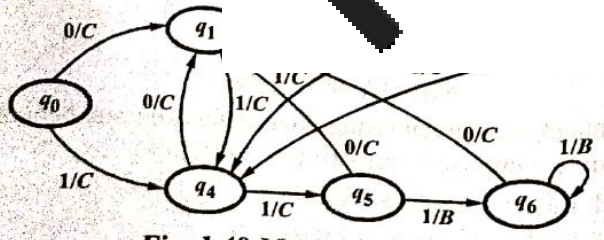
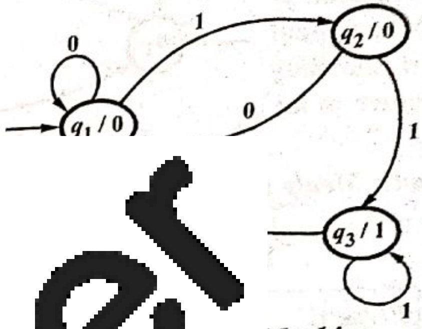


Fig. 1.48 Mealy Machine

Prob.31. Construct the Mealy machine equivalent to the given Moore machine –



Machine

agram into the transition table as

or Moore Machine

tate		Output
	a = *1*	
	q_2	0
	q_3	0
	q_3	1

ble as shown in table 1.33 by
e procedure is described in Q.17.

ransition Table

tate	
a = *1*	
State	*Output*
q_2	0
q_3	1
q_3	1

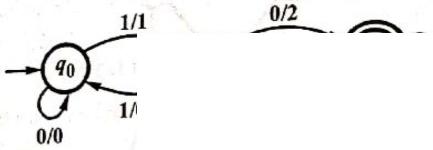
q_2 and q_3 are identical. So, we
q_3. We reconstruct the table by

the Given Moore Machine

Present State	*Next State*			
	a = *0*		*a* = *1*	
	State	*Output*	*State*	*Output*
→q_1	q_1	0	q_2	0
q_2	q_1	0	q_2	1

The pair of states and outputs in the next state column can be rearrang as given in table 1.39.

Prob.34. *Construct Moore machine for the following Mealy machin*



Sol. Transition table of th

State	
	Stat
→q_0	q_0
q_1	q_2
q_2	q_1

Rearranged transition table

Present State	
q_{00}	
q_{01}	
q_1	
q_2	

Resultant Moore machine i

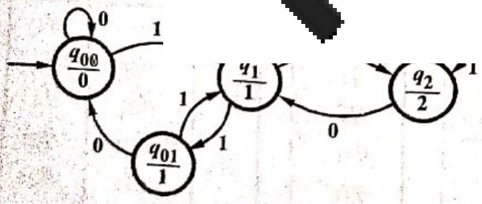


Fig. 1.54

17

Prob.35. *Convert the following Mealy machine into its equivalent Moore machine –*

Present	Next State	
	a = 1	
	tate	***Output***
	2	*n*
	2	*n*
	2	*y*

(R.G.P.V., Dec. 2012)

own in table 1.42.

ate Table

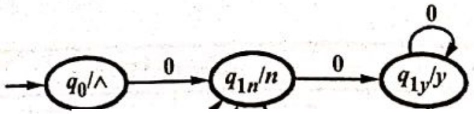
te	
a = 1	
te	***Output***
n	*n*
n	*n*
n	*n*
y	*y*
y	*y*

next state column can be is equivalent to the Moore

quivalent chine

			utput
			–
q_{1n}	q_{1y}	q_{2n}	*n*
q_{1y}	q_{1y}	q_{2n}	*y*
q_{2n}	q_{1n}	q_{2y}	*n*
q_{2y}	q_{1n}	q_{2y}	*y*

The corresponding transition diagram is shown in fig. 1.55.



Prob.36. Construct
machine –

Table 1.44

State
→ q_1
q_2
q_3
q_4

Sol. The transition
fig. 1.56.

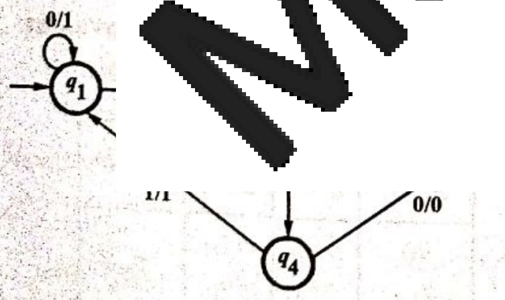


Fig. 1.56 Mealy Machine

Now, we construct the transition table as shown in table 1.45. Th
procedure is described in Q.16.

Table 1.45 Intermediate State Table

Present	*Next State*			
	a = 0		*a = 1*	
			State	*Output*
			q_{20}	0
			q_4	1
			q_4	1
			q_{31}	1
			q_{31}	1
			q_1	1

t state column can be rearranged

State Table

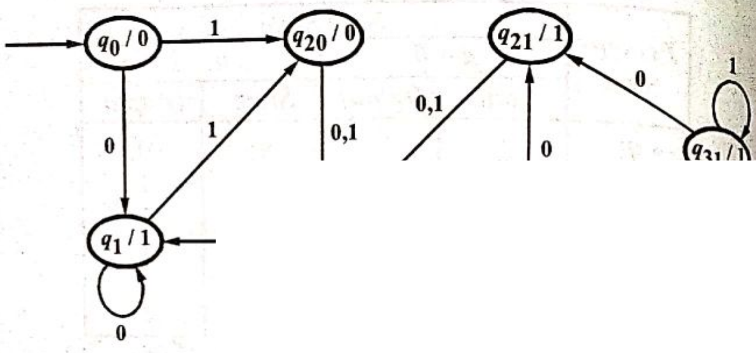
te *ı = 1*	*Output*
q_{20}	1
q_4	0
q_4	1
q_{31}	0
q_{31}	1
q_1	1

th output 1. This means that with
e starts at state q_1. To overcome
q_0, whose state transitions are
0. So, table 1.45 is transformed

o the Given Mealy Machine

		l	*Output*
			0
q_1	q_1	q_{20}	1
q_{20}	q_4	q_4	0
q_{21}	q_4	q_4	1
q_{30}	q_{21}	q_{31}	0
q_{31}	q_{21}	q_{31}	1
q_4	q_{30}	q_1	1

The corresponding transition diagram is shown in fig. 1.57.



UNIT

EXPRESSIONS, REGULAR
REGULAR GRAMMARS,
NERODE THEOREM,
JLAR LANGUAGES,
PING LEMMA

?

ıbols. In the case of a common
ıe alphabet to include 26 letters,
blanks and various punctuation
uage, we would want to add the

ence of symbols juxtaposed. For
s a string.

is the number of symbols in the
mpty string denoted by ∈, is the

ding symbols of that string and a
example, string *abc* has prefixes
: and *abc*. A prefix or suffix of a
proper prefix or suffix.

ınd v is the string obtained by
appending the symbols of v to the right end of w, that is, if

$$w = a_1a_2 \,.......a_n$$

and

$$v = b_1b_2 \,.......b_m$$

then the concatenation of w and v, denoted by wv, is

$$wv = a_1a_2 \,......a_n \; b_1b_2.....b_m$$

The empty string is the identity for the concatenation operator. Th $\in w = w \in = w$ for each string w. The ***reverse*** of a string is obtained by writin symbols in reverse order, if w is a string as shown above, then its reverse

$$w^R = a_n \ldots a_2 a_1$$

Q.3. Explain the ter

Ans. Refer to Q.1 ar

[redacted]

Q.4. What do you u

Explain the left line

Ans. A ***regular gram***

A grammar $G = (V,$

of the form –

where $A, B, \in V$, and $x \in$

A grammar is said to

Note that, in a regula side of any production. M leftmost or rightmost sym

Q.5. How can we co

Ans. We can constru following method –

(i) Construct a

(ii) Eliminate $\in$

(iii) From const terminal symbols and tran

Q.6. Describe regula regular expression.

Write short note on re

Define regular expressions.

(R.G.P.V., Dec. 20

Ans. The languages accepted by finite automata are easily described simple expressions called regular expressions. Regular expressions are use for representing certain sets of strings in an algebraic fashion. This notation regular expression involves a combination of strings of symbols from so alphabet over Σ, parenthesis, and the operators such as + and *.

Formal Definition of a Regular Expression – Regular expressions are constructed from primitive constituents by repeatedly applying certain recursive

iar arithmetic expressions are

d the sets that they denote are

ment of Σ), Λ, ϕ, are regular

expressions

ssions R_1 and R_2, written as

xpressions R_1 and R_2, written

ular expression R, written as

) is also a regular expression.

hy of operations as follows –

i.e., in evaluating a regular

perform iteration first, then

ion. ***(R.G.P.V., Dec. 2017)***

nd Q, are equivalent (i.e.,

ings.

essions. These are useful for

(xi) $(P + Q)^* = (P^*Q^*)^* = (P^* + Q^*)^*$

(xii) $(P + Q)R = PR + QR$ and $R(P + Q) = RP + RQ$.

('Set P' means that the set represented by the regular expression P)

Q.8. Define the language associated with regular expressions.

Ans. Regular expressions can be used to describe some simple languages. If R is a regular expression, we will let $L(R)$ denote the language associated with **R.** This language is defined as follows –

The language $L(R)$ denoted by any regular expression R is defined following rules –

(i) ϕ is a regular expression, denoting the empty set

(ii) Λ is a regular expression, denoting $\{\Lambda\}$

(iii) For every $a \in \Sigma$, ***a***

If R_1 and R_2 are regular ex

(iv) $L(R_1 + R_2) = L(R$

(vi) $L((R_1)) = L(R_1)$

The last four rules are us
recursively. While the first thı
recursion.

For example, we can exhib
follows –

$L(a^* (a + b)) = L($

$= (L$

$= \{$

$= \{$

But, there is a problem wi
define a language precisely if ***R***
ambiguity in dividing a complic
the regular expression ***ab*** + ***c***, m
find $L(ab + c) = \{ab, c\}$. But, th
taking $R_1 = a$ and $R_2 = b + c$. N
$ac\}$. To get overcome this, we req
but this makes the result cumb
from mathematics and progra
precedence rules for evaluation ir
and concatenation precedes uni

Q.9. Define regular set. E

Write short note on regula

Ans. A regular set is the set re
in $a, b \in \Sigma$, then a denotes the set $\{a\}$, $a + b$ denotes $\{a, b\}$, ab denotes
a^* denotes the set $\{\Lambda, a, aa, aaa, ...\}$ and $(a + b)^*$ denotes $\{a, b\}^*$.

For example, the following sets can be described as regular express

(i) $\{1\}$, $\{0\}$ are represented by **1** and **0**, respectively. 10
obtained by concatenating 1, 0 and 1. So, $\{101\}$ is represented by 101

(ii) $\{abba\}$ is represented by ***abba***

(iii) As $\{ab, ba\}$ is the union of $\{ab\}$ and $\{ba\}$, $\{ab, ba\}$ is repre-
ented by ***ab*** + ***ba***

(iv) The set $\{\Lambda, 10\}$ is represented by $\boldsymbol{\Lambda + 10}$

(v) The set $\{abb, a, b, bba\}$ is represented by ***abb*** + ***a*** + ***b*** + ***bba***

$\}^*$, it is represented by a^*

$\}$ can be obtained by

$)^*$ represents $\{1, 11, 111,$

regular expressions ?

Q. P and Q are equivalent
are equivalent if and only

three ways –

ne. (For non-equivalence,

ence of P and Q
mata M and M' and prove
ve that M and M' are not

according to the problem.

ng the relation between

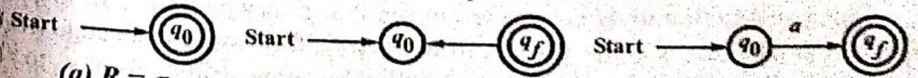
re precisely the languages
NFA with $\in$-transitions

etween transition systems

there exists an NFA with

f operators in the regular
having one final state and

The expression R must be $\in$, ϕ, or a for some a in Σ. The NFAs
n fig. 2.1 (a), (b) and (c) clearly satisfy the conditions.



(a) $R = \in$ (b) $R = \phi$ (c) $R = a$

Fig. 2.1 Transition Systems Recognizing Elementary Regular Sets

Induction Step – Assume that the theorem is true for regular expre
with fewer than i operators, $i \geq 1$. Let $\boldsymbol{R}$ have i operators. There are
cases depending on the form of $\boldsymbol{R}$.

(i) *Case 1* – For union, i.e., $\boldsymbol{R} = \boldsymbol{R_1} + \boldsymbol{R_2}$. Both $\boldsymbol{R_1}$ and $\boldsymbol{R}$
have fewer than i operators. Thus there are NFAs $M_1 = (Q$
and $M_2 = (Q_2, \Sigma_2, \delta_2, q_2,$
Since, we may rename states
disjoint. Let q_0 be a new initi

$M =$

where δ is defined by,

(a) $\delta(q_0, \in) =$

(b) $\delta(q, a) =$

(c) $\delta(q, a) =$

(d) $\delta(f_1, \in) =$

Recall by the inductiv
f_1 or f_2 in M_1 or M_2. Thus a
construction of M is depi
labelled x in M from q_0 to
from q_1 to f_1 or a path in
as desired.

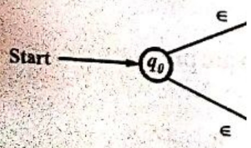


Fig. 2.2 Trans

(ii) *Case 2* – Fo
in case 1 and then we co

M

where δ is given by,

(a) $\delta(q, a)$ q in $Q_1 - \{f_1\}$ and a in

(b) $\delta(f_1, \in) = q_2$

(c) $\delta(q, a) = \delta_2(q, a)$ for q in Q_2 and a in $\Sigma_2 \cup \{\in\}$

The construction of M is depicted in fig. 2.3. Every path in M from
f_2 is a path labelled by some string x from q_1 to f_1, followed by the edge
f_1 to q_2 labelled $\in$, followed by a path labelled by some string y from
Thus $L(M) = L(M_1)\ L(M_2)$ as desired.

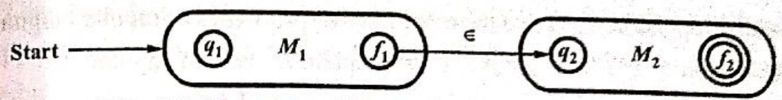
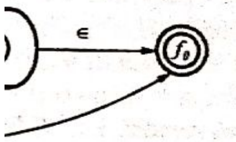


Fig. 2.3 Transition System Recognizing $R = R_1R_2$

Let $M_1 = (Q_1, \Sigma_1, \delta_1, q_1,$

$q_0, \{f_0\}$

$\{f_1\}$ and a in $\Sigma_1 \cup \{\in\}$
It follows that there is a
an write $x = x_1, x_2 \ldots x_j$ for
that each x_i is in $L(M_1)$



ing $R = R_1^*$

erting a regular expression

lar sets.

intersection.
3, Dec. 2003, June 2004)

under complement and
(R.G.P.V., Dec. 2002)

lar grammar.
(R.G.P.V., Dec. 2017)

Mention the closure properties of regular languages. (R.G.P.V., Nov. 2018)

Ans. If L_1 and L_2 are regular languages, then there exist regular expression
and $\boldsymbol{R_2}$ such that $L_1 = L(R_1)$ and $L_2 = L(R_2)$. By definition,
$+ \boldsymbol{R_2}, \boldsymbol{R_1R_2}$, and $\boldsymbol{R_1}^*$ are regular expressions denoting the languages
$\cup L_2, L_1L_2$, and L_1^*, respectively. Thus, closure under union,
oncatenation, and star-closure is immediate.

Following theorems prove the other closure properties of regular lang

Theorem 1. If L is regular then transpose L^T is also regular.

Proof. Every regular expression is recognized by transition syst
transition system M can be converted into finite accepter (FA), accepti
same set as M. Any set accepted by
A subset L of Σ^* is regular set if and
We can construct a FA $M = (Q,$
$T(M) =$

We construct a transition syst
M and reversing the direction of th
M' is defined as the set F, and q_0 is
$M' = (Q, \Sigma, \delta', F, \{q_0\})$.

If $w \in T(M)$, we have a path
value w. By 'reversing the edges',
F to q_0. Its path value is w^T. So w
if $w_1 \in T(M')$, then $w_1{}^T \in T(M)$. T
that $T(M') = T(M)^T$. We can prove
$T(M')$ by induction on $|w|$. So
L of Σ^* is a regular set if and only
$T(M')$ is regular, i.e., $T(M)^T$ is reg

[redacted]

Theorem 4. The class of regular sets is closed under substitution.

Proof. Let $R \subseteq \Sigma^*$ be a regular set and for each a in Σ let $R_a \subseteq \Delta^*$ be
regular set. Let $f : \Sigma \rightarrow \Delta^*$ be the substitution defined by $f(a) = R_a$. Select
expressions denoting R and each R_a. Replace each occurrence of the
gular expression for R_a. To
tes $f(R)$, observe that the
nion, product or closure of
of operators in the regular

ed under homomorphisms

s immediately from closure
s a substitution, in which

m, let $M = (Q, \Sigma, \delta, q_0, F)$
rphism from Δ to Σ^*. We
ymbol a in Δ and simulating
l define $\delta'(q, a)$, for q in Q
a long string, or $\in$, but δ is

[redacted]

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer

Msinventer